Computer Architecture
Project 1, Datapath Design, Part B

Joanna Whittam (15319524)

March 22nd 2017

1 Register file

1.1 Reg

1.1.1 Reg Code

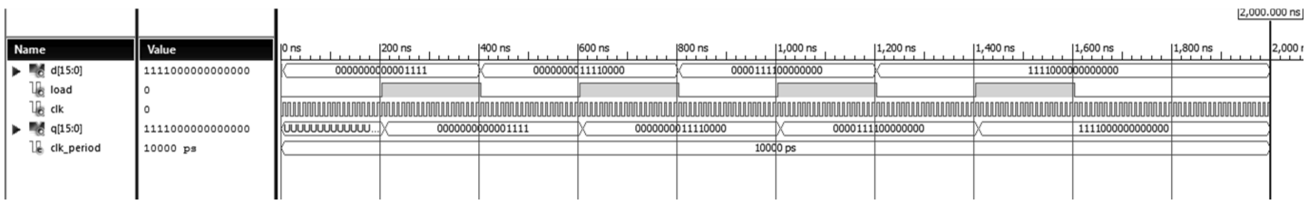
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity reg is
        Port ( D: in STD_LOGIC_VECTOR (15 downto 0);
                load, CLK: in STD_LOGIC;
                Q: out STD_LOGIC_VECTOR (15 downto 0) );
end reg;

architecture Behavioral of reg is

begin
        process(CLK)
        begin
                if (rising_edge(CLK)) then
                        if load='1' then
                                Q <= D after 5 ns;
                        end if;
                end if;
        end process;
end Behavioral;
```

1.1.2 Reg Test Bench

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY reg_tb IS
END reg_tb;

ARCHITECTURE behavior OF reg_tb IS

          -- Component Declaration for the Unit Under Test (UUT)

          COMPONENT reg
          PORT( D: IN  std_logic_vector(15 downto 0);
                     load, CLK: IN  std_logic;
                     Q: OUT  std_logic_vector(15 downto 0) );
          END COMPONENT;


         --Inputs
         signal D: std_logic_vector(15 downto 0) := (others => '0');
         signal load: std_logic := '0';
         signal CLK: std_logic := '0';

         --Outputs
         signal Q: std_logic_vector(15 downto 0);
```

```
            -- Clock period definitions
            constant CLK_period: time := 10 ns;

BEGIN
            -- Instantiate the Unit Under Test (UUT)
            uut: reg PORT MAP (
                    D => D,
                    load => load,
                    CLK => CLK,
                    Q => Q
            );

            -- Clock process definitions
            CLK_process :process
            begin
                    CLK <= '0';
                    wait for CLK_period/2;
                    CLK <= '1';
                    wait for CLK_period/2;
            end process;

          -- Stimulus process
            stim_proc: process
            begin
                            -- hold reset state for 50 ns.
                            wait for 5 ns;

                            D <= "0000000000001111";
                            wait for CLK_period*20;
                            load <= '1';
                            wait for CLK_period*20;
                            load <= '0';

                            D <= "0000000011110000";
                            wait for CLK_period*20;
                            load <= '1';
                            wait for CLK_period*20;
                            load <= '0';

                            D <= "0000111100000000";
                            wait for CLK_period*20;
                            load <= '1';
                            wait for CLK_period*20;
                            load <= '0';

                            D <= "1111000000000000";
                            wait for CLK_period*20;
                            load <= '1';
                            wait for CLK_period*20;
                            load <= ‘0';
                            wait;
            end process;
END;
```

1.1.3 Reg Test Results



1.2 Decoder

1.2.1 Decoder Code

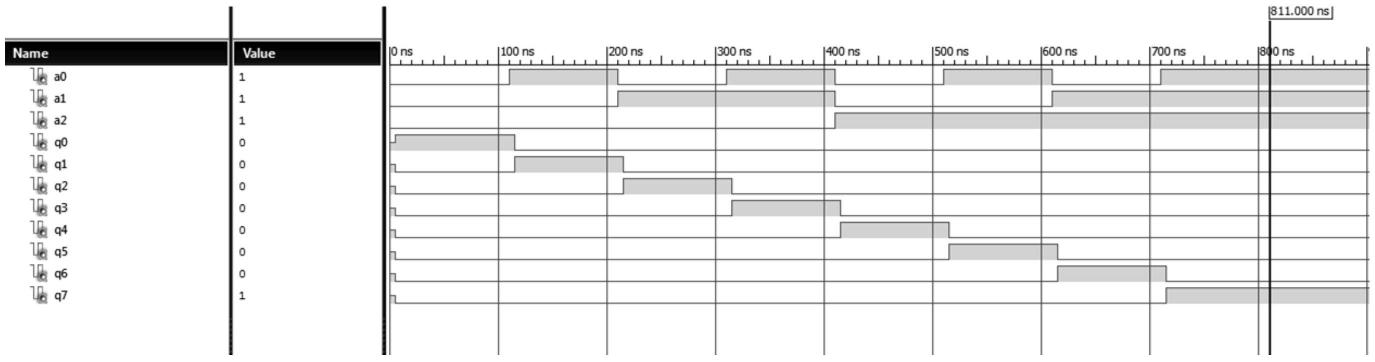
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity decoder_3to8 is
    Port ( A0,A1,A2: in  STD_LOGIC;
           Q0,Q1,Q2,Q3,Q4,Q5,Q6,Q7: out  STD_LOGIC );
end decoder_3to8;

architecture Behavioral of decoder_3to8 is

begin
        Q0 <=   ((not A2)       and (not A1)    and (not A0))   after 1 ns;
        Q1 <=   ((not A2)       and (not A1)    and  A0)        after 1 ns;
        Q2 <=    ((not A2)      and  A1         and (not A0))   after 1 ns;
        Q3 <=    ((not A2)      and  A1         and  A0)        after 1 ns;
        Q4 <=    ( A2           and (not A1)    and (not A0))   after 1 ns;
        Q1 <=    ( A2           and (not A1)    and  A0)        after 1 ns;
        Q6 <=    ( A2           and  A1         and (not A0))   after 1 ns;
        Q7 <=    ( A2           and  A1         and  A0)        after 1 ns;
end Behavioral;
```

1.2.2 Decoder Test Bench

```
        LIBRARY ieee;
        USE ieee.std_logic_1164.ALL;

        ENTITY decoder_3to8_tb IS
        END decoder_3to8_tb;

        ARCHITECTURE behavior OF decoder_3to8_tb IS

                  -- Component Declaration for the Unit Under Test (UUT)
                  COMPONENT decoder_3to8
                  PORT(         A0,A1,A2: in  STD_LOGIC;
                                Q0,Q1,Q2,Q3,Q4,Q5,Q6,Q7: out  STD_LOGIC );
                  END COMPONENT;

                 --Inputs
                 signal A0 : std_logic := '0';
                 signal A1 : std_logic := '0';
                 signal A2 : std_logic := '0';

                --Outputs
                signal Q0,Q1,Q2,Q3,Q4,Q5,Q6,Q7: std_logic;

BEGIN
                -- Instantiate the Unit Under Test (UUT)
                uut: decoder_3to8 PORT MAP (
                        A0 => A0,
                        A1 => A1,
                        A2 => A2,
                        Q0 => Q0,
                        Q1 => Q1,
                        Q2 => Q2,
                        Q3 => Q3,
                        Q4 => Q4,
                        Q5 => Q5,
                        Q6 => Q6,
                        Q7 => Q7
                 );
```

```
            -- Stimulus process
            stim_proc: process
            begin
                    -- hold reset state for 10 ns.
                    wait for 10 ns;

                    -- 0
                    A0<='0'; A1<='0'; A2<='0';
                    wait for 100 ns;

                    -- 1
                    A0<='1'; A1<='0'; A2<='0';
                    wait for 100 ns;

                    -- 2
                    A0<='0'; A1<='1'; A2<='0';
                    wait for 100 ns;

                    -- 3
                    A0<='1'; A1<='1'; A2<='0';
                    wait for 100 ns;

                    -- 4
                    A0<='0'; A1<='0'; A2<='1';
                    wait for 100 ns;

                    -- 5
                    A0<='1'; A1<='0'; A2<='1';
                    wait for 100 ns;

                    -- 6
                    A0<='0'; A1<='1'; A2<='1';
                    wait for 100 ns;

                    -- 7
                    A0<='1'; A1<='1'; A2<='1';
                    wait;
           end process;
   END;
```

1.2.3 Decoder Test Results



1.3 Mux8_16bit

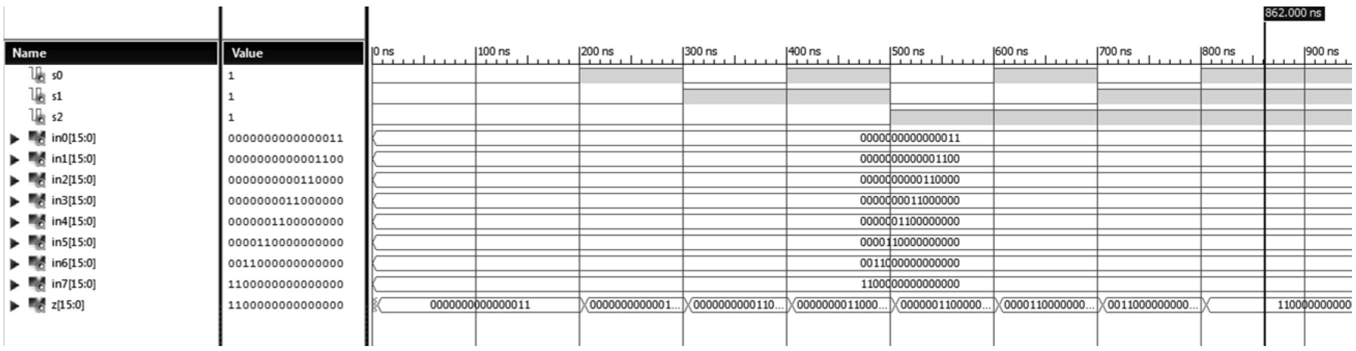
1.3.1 Mux8_16bit Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity mux8_16bit is
        Port ( S0,S1,S2: in STD_LOGIC;
              In0,In1,In2,In3,In4,In5,In6,In7: in STD_LOGIC_VECTOR (15 downto 0);
              Z: out STD_LOGIC_VECTOR (15 downto 0));
end mux8_16bit;

architecture Behavioral of mux8_16bit is
begin
Z <=    In0 after 5 ns when S2='0' and S1='0' and S0='0' else
                In1 after 5 ns when S2='0' and S1='0' and S0='1' else
                In2 after 5 ns when S2='0' and S1='1' and S0='0' else
                In3 after 5 ns when S2='0' and S1='1' and S0='1' else
                In4 after 5 ns when S2='1' and S1='0' and S0='0' else
                In5 after 5 ns when S2='1' and S1='0' and S0='1' else
                In6 after 5 ns when S2='1' and S1='1' and S0='0' else
                In7 after 5 ns when S2='1' and S1='1' and S0='1';
end Behavioral;
```

1.3.2 Mux8_16bit Test Bench

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY mux8_16bit_tb IS
END mux8_16bit_tb;

ARCHITECTURE behavior OF mux8_16bit_tb IS
        -- Component Declaration for the Unit Under Test (UUT)
        COMPONENT mux8_16bit
        PORT(  S0,S1,S2: in  STD_LOGIC;
                 In0,In1,In2,In3,In4,In5,In6,In7: in  STD_LOGIC_VECTOR (15 downto 0);
                 Z: out STD_LOGIC_VECTOR (15 downto 0));
        END COMPONENT;

        --Inputs
        signal S0 : std_logic := '0';
        signal S1 : std_logic := '0';
        signal S2 : std_logic := '0';
        signal In0 : std_logic_vector(15 downto 0) := (others => '0');
        signal In1 : std_logic_vector(15 downto 0) := (others => '0');
        signal In2 : std_logic_vector(15 downto 0) := (others => '0');
        signal In3 : std_logic_vector(15 downto 0) := (others => '0');
        signal In4 : std_logic_vector(15 downto 0) := (others => '0');
        signal In5 : std_logic_vector(15 downto 0) := (others => '0');
        signal In6 : std_logic_vector(15 downto 0) := (others => '0');
        signal In7 : std_logic_vector(15 downto 0) := (others => '0');
        signal Z : std_logic_vector(15 downto 0) := (others => '0');

BEGIN
        -- Instantiate the Unit Under Test (UUT)
        uut: mux8_16bit PORT MAP (
                S0 => S0,
                S1 => S1,
                S2 => S2,
                In0 => In0,
                In1 => In1,
			In2 => In2,
			In3 => In3,
			In4 => In4,
			In5 => In5,
			In6 => In6,
			In7 => In7,
			Z => Z
		);

		-- Stimulus process
		stim_proc: process
		begin
			In0 <= "0000000000000011";
			In1 <= "0000000000001100";
			In2 <= "0000000000110000";
			In3 <= "0000000011000000";
			In4 <= "0000001100000000";
			In5 <= "0000110000000000";
			In6 <= "0011000000000000";
			In7 <= "1100000000000000";
			 wait for 100 ns;

			-- 0
			S0<='0'; S1<='0'; S2<='0';
			wait for 100 ns;

			-- 1
			S0<='1'; S1<='0'; S2<='0';
			wait for 100 ns;

			-- 2
			S0<= '0'; S1<='1'; S2<='0';
			wait for 100 ns;

			-- 3
			S0<= '1'; S1<='1'; S2<='0';
			wait for 100 ns;
			-- 4
			S0<= '0'; S1<='0'; S2<='1';
			wait for 100 ns;

			-- 5
			S0<= '1'; S1<='0'; S2<='1';
			wait for 100 ns;

			-- 6
			S0<= '0'; S1<='1'; S2<='1';
			wait for 100 ns;

			-- 7
			S0<='1'; S1<='1'; S2<='1';
			wait;
		end process;
	END;
```

1.3.3 Mux8_16bit Test Results


862.000 ns
Name
Value
s0 1
s1 1
s2 1
in0[15:0] 0000000000000011
in1[15:0] 0000000000001100
in2[15:0] 0000000000110000
in3[15:0] 0000000011000000
in4[15:0] 0000001100000000
in5[15:0] 0000110000000000
in6[15:0] 0011000000000000
in7[15:0] 1100000000000000
z[15:0] 1100000000000000
0 ns
100 ns
200 ns
300 ns
400 ns
500 ns
600 ns
700 ns
800 ns
900 ns

1.4 Register File

1.4.1 Register_file Code

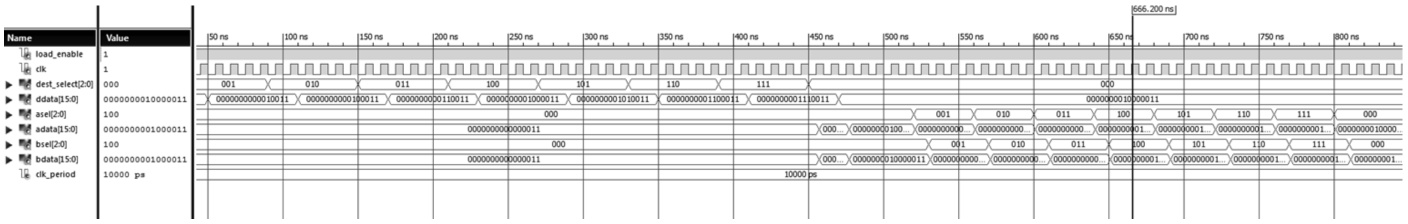
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity register_file_v2 is
    PORT(  load_enable, CLK : in  STD_LOGIC;
          dest_select, Asel, Bsel : in  STD_LOGIC_VECTOR(2 downto 0);
          DData : in  STD_LOGIC_VECTOR (15 downto 0);
          AData, BData : out  STD_LOGIC_VECTOR (15 downto 0));
end register_file_v2;

architecture Behavioral of register_file_v2 is
--COMPONENTS
            component decoder_3to8 is
   Port (  A0,A1,A2: in  STD_LOGIC;
          Q0,Q1,Q2,Q3,Q4,Q5,Q6,Q7: out  STD_LOGIC );
            end component;

            component reg
   Port (  D: in STD_LOGIC_VECTOR (15 downto 0);
          load, CLK: in STD_LOGIC;
          Q: out STD_LOGIC_VECTOR (15 downto 0) );
            end component;

            component mux8_16bit is
   Port (  S0,S1,S2: in STD_LOGIC;
          In0,In1,In2,In3,In4,In5,In6,In7: in STD_LOGIC_VECTOR (15 downto 0);
          Z: out STD_LOGIC_VECTOR (15 downto 0));
            end component;

--SIGNALS
signal load_r0,load_r1,load_r2,load_r3,load_r4,load_r5,load_r6,load_r7: std_logic;
signal decode0_q, decode1_q, decode2_q, decode3_q, decode4_q, decode5_q, decode6_q,
            decode7_q: std_logic;
signal reg0_q,reg1_q,reg2_q,reg3_q,reg4_q,reg5_q,reg6_q,reg7_q: std_logic_vector(15 downto 0);

begin
            --Decoder
            decoder: decoder_3to8 PORT MAP(
                        A0 => dest_select(0),
                        A1 => dest_select(1),
                        A2 => dest_select(2),
                        Q0 => decode0_q,
                        Q1 => decode1_q,
                        Q2 => decode2_q,
			Q3 => decode3_q,
			Q4 => decode4_q,
			Q5 => decode5_q,
			Q6 => decode6_q,
			Q7 => decode7_q
		);

	load_r0 <= load_enable and decode0_q;
	load_r1 <= load_enable and decode1_q;
	load_r2 <= load_enable and decode2_q;
	load_r3 <= load_enable and decode3_q;
	load_r4 <= load_enable and decode4_q;
	load_r5 <= load_enable and decode5_q;
	load_r6 <= load_enable and decode6_q;
	load_r7 <= load_enable and decode7_q;

		reg00: reg PORT MAP(
			D => DData,
			load => load_r0,
			Clk => Clk,
			Q => reg0_q
		);

		reg01: reg PORT MAP(
			D => DData,
			load => load_r1,
			Clk => Clk,
			Q => reg1_q
		);

		reg02: reg PORT MAP(
			D => DData,
			load => load_r2,
			Clk => Clk,
			Q => reg2_q
		);

		reg03: reg PORT MAP(
			D => DData,
			load => load_r3,
			Clk => Clk,
			Q => reg3_q
		);

		reg04: reg PORT MAP(
			D => DData,
			load => load_r4,
			Clk => Clk,
			Q => reg4_q
		);

		reg05: reg PORT MAP(
			D => DData,
			load => load_r5,
			Clk => Clk,
			Q => reg5_q
		);

		reg06: reg PORT MAP(
			D => DData,
			load => load_r6,
			Clk => Clk,
			Q => reg6_q
		);

		reg07: reg PORT MAP(
                D => DData,
                load => load_r7,
                Clk => Clk,
                Q => reg7_q
        );

        A_mux8_16bit: mux8_16bit PORT MAP(
                S0 => Asel(0),
                S1 => Asel(1),
                S2 => Asel(2),
                In0 => reg0_q,
                In1 => reg1_q,
                In2 => reg2_q,
                In3 => reg3_q,
                In4 => reg4_q,
                In5 => reg5_q,
                In6 => reg6_q,
                In7 => reg7_q,
                Z => AData
        );

        B_mux8_16bit: mux8_16bit PORT MAP(
                S0 => Bsel(0),
                S1 => Bsel(1),
                S2 => Bsel(2),
                In0 => reg0_q,
                In1 => reg1_q,
                In2 => reg2_q,
                In3 => reg3_q,
                In4 => reg4_q,
                In5 => reg5_q,
                In6 => reg6_q,
                In7 => reg7_q,
                Z => BData
        );

end Behavioral;
```

1.4.2 Register_file Test Bench

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY register_file_v2_tb IS
END register_file_v2_tb;

ARCHITECTURE behavior OF register_file_v2_tb IS

    COMPONENT register_file_v2
    PORT(  load_enable, CLK : in  STD_LOGIC;
           dest_select, Asel, Bsel : in  STD_LOGIC_VECTOR(2 downto 0);
           DData : in  STD_LOGIC_VECTOR (15 downto 0);
           AData, BData : out  STD_LOGIC_VECTOR (15 downto 0));
    END COMPONENT;

   --Inputs
   signal load_enable: std_logic := '0';
        signal CLK: std_logic:= '0';
   signal dest_select: std_logic_vector(2 downto 0) := (others => '0');
   signal Asel, Bsel: std_logic_vector(2 downto 0) := (others => '0');
   signal DData: std_logic_vector(15 downto 0) := (others => '0');
        --Outputs
   signal AData, BData : std_logic_vector(15 downto 0);
```

```
          -- Clock period definitions
   constant CLK_period: time := 10 ns;

BEGIN
   uut: register_file_v2 PORT MAP (
          load_enable => load_enable,
                                    CLK => CLK,
          dest_select => dest_select,
          Asel => Asel,
          Bsel => Bsel,
          DData => DData,
          AData => AData,
          BData => BData
        );

   CLK_process :process
   begin
                        CLK <= '0';
                        wait for CLK_period/2;
                        CLK <= '1';
                        wait for CLK_period/2;
   end process;

   stim_proc: process
   begin

      wait for 10 ns;
                        load_enable <= '1';
                        DData <= "0000000000000011";
                        dest_select <= "000";

                        --set registers to unique values
                        for i in 0 to 7 loop
                                    wait for CLK_period*2;
                                    dest_select <= STD_LOGIC_VECTOR (unsigned(dest_select) + 1);
                                    wait for CLK_period*2;
                                    DData <= STD_LOGIC_VECTOR (unsigned(DData) +16);
                                    wait for CLK_period*2;
                        end loop;

                        --read each register from A and B outputs
                        Asel <= "000";
                        Bsel <= "000";
                        for i in 0 to 7 loop
                                    wait for CLK_period*3;
                                    Asel <= STD_LOGIC_VECTOR (unsigned(Asel) + 1);
                                    wait for CLK_period;
                                    Bsel <= STD_LOGIC_VECTOR (unsigned(Bsel) + 1);
                        end loop;

      wait;
   end process;

END;
```

1.4.2 Register_file Test Results



2 Function Unit

2.1 Zero Detect

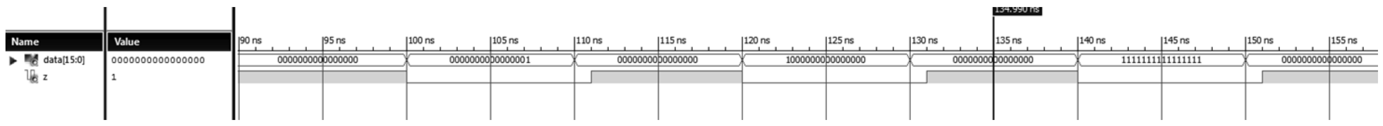
2.1.1 Zero_detect code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity zero_detect_16 is
        Port (   data : in  STD_LOGIC_VECTOR (15 downto 0);
                 z : out  STD_LOGIC);
end zero_detect_16;

architecture Behavioral of zero_detect_16 is

begin
z     <=    '1' after 1 ns when data="0000000000000000"
        else '0';

end Behavioral;
```

2.1.2 Zero_detect Test Bench

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY zero_detect_tb IS
END zero_detect_tb;

ARCHITECTURE behavior OF zero_detect_tb IS

-- Component Declaration for the Unit Under Test (UUT)
COMPONENT zero_detect_16
        PORT(
                data : IN  std_logic_vector(15 downto 0);
                z : OUT  std_logic
        );
END COMPONENT;


--Inputs
 signal data : std_logic_vector(15 downto 0) := (others => '0');

--Outputs
signal z : std_logic;

BEGIN
        -- Instantiate the Unit Under Test (UUT)
        uut: zero_detect_16 PORT MAP (
                data => data,
                z => z
        );

        -- Stimulus process
        stim_proc: process
        begin
        wait for 100 ns;
                data <= "0000000000000001";
                wait for 10 ns;

                data <= "0000000000000000";
                wait for 10 ns;

                data <= "1000000000000000";
                wait for 10 ns;

                data <= "0000000000000000";
                wait for 10 ns;

                data <= "1111111111111111";
                wait for 10 ns;

                data <= "0000000000000000";
                wait for 10 ns;
                Wait;
        end process;
END;
```

2.1.3 Zero_detect Test Results



2.2 Mux2_16bit

In my file this entity is incorrectly named as mux2_4bit. This will be changed in later versions of the code.

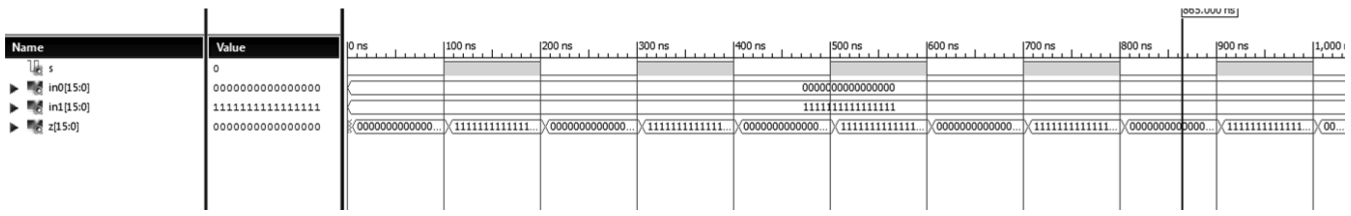
2.2.1 Mux2_16bit Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity mux2_16bit is
        Port ( S: in  STD_LOGIC;
                In0, In1: in  STD_LOGIC_VECTOR (15 downto 0);
                Z: out  STD_LOGIC_VECTOR (15 downto 0));
end mux2_16bit;

architecture Behavioral of mux2_16bit is

begin
Z    <=    In0 after 1 ns when S='0' else
           In1 after 1 ns when S='1';
end Behavioral;
```

2.2.2 Mux2_16bit Test Bench

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY mux2_4bit_tb IS
END mux2_4bit_tb;

ARCHITECTURE behavior OF mux2_4bit_tb IS
          COMPONENT mux2_4bit
          PORT(     S: in  STD_LOGIC;
                In0, In1: in  STD_LOGIC_VECTOR (15 downto 0);
                Z: out  STD_LOGIC_VECTOR (15 downto 0)
          );
          END COMPONENT;

         --Inputs
         signal S : std_logic := '0';
         signal In0 : std_logic_vector(15 downto 0) := (others => '0');
         signal In1 : std_logic_vector(15 downto 0) := (others => '0');

         --Outputs
         signal Z : std_logic_vector(15 downto 0);

BEGIN
        -- Instantiate the Unit Under Test (UUT)
        uut: mux2_bit PORT MAP (
                S => S,
                In0 => In0,
                In1 => In1,
                Z => Z
        );

         -- Stimulus process
         stim_proc: process
         begin
                In0 <= "0000000000000000";
                In1 <= "1111111111111111";

                S <= '0';
                wait for 100 ns;

                S <= '1';
                wait for 100 ns;
       end process;
END;
```

2.2.3 Mux2_16bit Test Results



2.3 Mux16_1bit

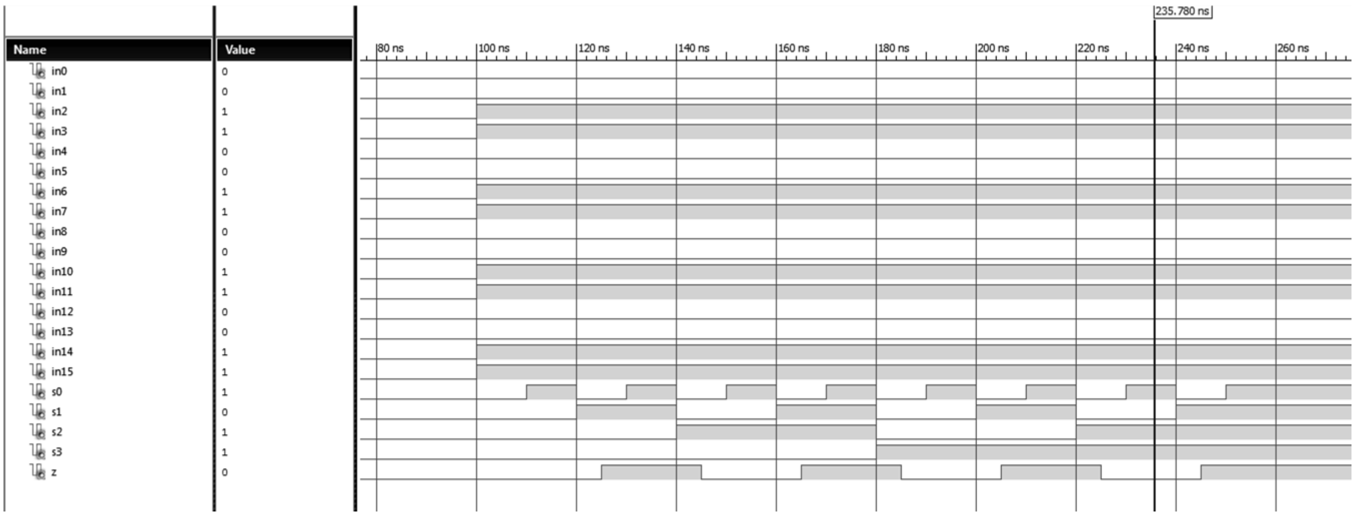
2.3.1 Mux16_1bit Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity mux16_1bit is
    Port ( in0,in1,in2,in3,in4,in5,in6,in7,in8,in9,in10,in11,in12,in13,in14,in15: in STD_LOGIC;
          s0,s1,s2,s3: in STD_LOGIC;
          z: out STD_LOGIC);
end mux16_1bit;

architecture Behavioral of mux16_1bit is

begin
Z <=     in0  after 5 ns     when s3='0'     and s2='0'     and s1='0'     and s0='0'     else
         in1  after 5 ns     when s3='0'     and s2='0'     and s1='0'     and s0='1'     else
         in2  after 5 ns     when s3='0'     and s2='0'     and s1='1'     and s0='0'     else
         in3  after 5 ns     when s3='0'     and s2='0'     and s1='1'     and s0='1'     else
         in4  after 5 ns     when s3='0'     and s2='1'     and s1='0'     and s0='0'     else
         in5  after 5 ns     when s3='0'     and s2='1'     and s1='0'     and s0='1'     else
         in6  after 5 ns     when s3='0'     and s2='1'     and s1='1'     and s0='0'     else
         in7  after 5 ns     when s3='0'     and s2='1'     and s1='1'     and s0='1'     else
         in8  after 5 ns     when s3='1'     and s2='0'     and s1='0'     and s0='0'     else
         in9  after 5 ns     when s3='1'     and s2='0'     and s1='0'     and s0='1'     else
         in10 after 5 ns     when s3='1'     and s2='0'     and s1='1'     and s0='0'     else
         in11 after 5 ns     when s3='1'     and s2='0'     and s1='1'     and s0='1'     else
         in12 after 5 ns     when s3='1'     and s2='1'     and s1='0'     and s0='0'     else
         in13 after 5 ns     when s3='1'     and s2='1'     and s1='0'     and s0='1'     else
         in14 after 5 ns     when s3='1'     and s2='1'     and s1='1'     and s0='0'     else
         in15 after 5 ns     when s3='1'     and s2='1'     and s1='1'     and s0='1';

end Behavioral;
```

2.3.2 Mux16_1bit Test Bench

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY mux16_1bit_tb IS
END mux16_1bit_tb;

ARCHITECTURE behavior OF mux16_1bit_tb IS

 COMPONENT mux16_1bit
         PORT(    in0,in1,in2,in3,in4,in5,in6,in7,in8,in9,in10,in11,in12,in13,in14,in15: in STD_LOGIC;
                  s0,s1,s2,s3: in STD_LOGIC;
                  z : out  std_logic );
 END COMPONENT;

  --Inputs
  signal in0,in1,in2,in3,in4,in5,in6,in7,in8,in9,in10,in11,in12,in13,in14,in15 : std_logic := '0';
  signal s0,s1,s2,s3 : std_logic := '0';

         --Outputs
  signal z : std_logic;

BEGIN
  uut: mux16_1bit PORT MAP (
        in0 => in0,
        in1 => in1,
        in2 => in2,
        in3 => in3,
        in4 => in4,
        in5 => in5,
        in6 => in6,
        in7 => in7,
        in8 => in8,
        in9 => in9,
        in10 => in10,
        in11 => in11,
        in12 => in12,
        in13 => in13,
          in14 => in14,
          in15 => in15,
          s0 => s0,
          s1 => s1,
          s2 => s2,
          s3 => s3,
          z => z
        );

   stim_proc: process
   begin
               wait for 100 ns;
              in0 <= '0';
              in1 <= '0';
              in2 <= '1';
              in3 <= '1';
              in4 <= '0';
              in5 <= '0';
              in6 <= '1';
              in7 <= '1';
              in8 <= '0';
              in9 <= '0';
              in10 <= '1';
              in11 <= '1';
              in12 <= '0';
              in13 <= '0';
              in14 <= '1';
              in15 <= '1';

              s0 <= '0';      s1 <= '0';      s2 <= '0';      s3 <= '0';      wait for 10 ns;
              s0 <= '1';      s1 <= '0';      s2 <= '0';      s3 <= '0';      wait for 10 ns;
              s0 <= '0';      s1 <= '1';      s2 <= '0';      s3 <= '0';      wait for 10 ns;
              s0 <= '1';      s1 <= '1';      s2 <= '0';      s3 <= '0';      wait for 10 ns;
              s0 <= '0';      s1 <= '0';      s2 <= '1';      s3 <= '0';      wait for 10 ns;
              s0 <= '1';      s1 <= '0';      s2 <= '1';      s3 <= '0';      wait for 10 ns;
              s0 <= '0';      s1 <= '1';      s2 <= '1';      s3 <= '0';      wait for 10 ns;
              s0 <= '1';      s1 <= '1';      s2 <= '1';      s3 <= '0';      wait for 10 ns;
              s0 <= '0';      s1 <= '0';      s2 <= '0';      s3 <= '1';      wait for 10 ns;
              s0 <= '1';      s1 <= '0';      s2 <= '0';      s3 <= '1';      wait for 10 ns;
              s0 <= '0';      s1 <= '1';      s2 <= '0';      s3 <= '1';      wait for 10 ns;
              s0 <= '1';      s1 <= '1';      s2 <= '0';      s3 <= '1';      wait for 10 ns;
              s0 <= '0';      s1 <= '0';      s2 <= '1';      s3 <= '1';      wait for 10 ns;
              s0 <= '1';      s1 <= '0';      s2 <= '1';      s3 <= '1';      wait for 10 ns;
              s0 <= '0';      s1 <= '1';      s2 <= '1';      s3 <= '1';      wait for 10 ns;
              s0 <= '1';      s1 <= '1';      s2 <= '1';      s3 <= '1';      wait for 10 ns;
              wait;
end process;

END;
```

2.3.3 Mux16_1bit Test Results



2.4 Shifter_16bit

2.4.1 Shifter_16bit Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity shifter_16bit is
    Port ( input: in STD_LOGIC_VECTOR(15 downto 0);
           sel0,sel1,sel2,sel3 : in  STD_LOGIC;
           y: out STD_LOGIC_VECTOR(15 downto 0) );
end shifter_16bit;

architecture Behavioral of shifter_16bit is
--COMPONENTS
        component mux16_1bit is
        Port ( in0,in1,in2,in3,in4,in5,in6,in7,in8,in9,in10,in11,in12,in13,in14,in15: in STD_LOGIC;
                        s0,s1,s2,s3: in STD_LOGIC;
                        z: out STD_LOGIC );
        end component;

begin
-- PORT MAPS
        mux0: mux16_1bit port map (
                in0 => input(0),
                in1 => input(15),
                in2 => input(14),
                in3 => input(13),
                in4 => input(12),
                in5 => input(11),
                in6 => input(10),
                in7 => input(9),
                in8 => input(8),
                in9 => input(7),
                in10 => input(6),
                in11 => input(5),
                in12 => input(4),
                in13 => input(3),
                in14 => input(2),
                in15 => input(1),
                s0 => sel0,
                s1 => sel1,
        s2 => sel2,
        s3 => sel3,
        z => y(0)
);

mux1: mux16_1bit port map (
        in0 => input(1),
        in1 => input(0),
        in2 => input(15),
        in3 => input(14),
        in4 => input(13),
        in5 => input(12),
        in6 => input(11),
        in7 => input(10),
        in8 => input(9),
        in9 => input(8),
        in10 => input(7),
        in11 => input(6),
        in12 => input(5),
        in13 => input(4),
        in14 => input(3),
        in15 => input(2),
        s0 => sel0,
        s1 => sel1,
        s2 => sel2,
        s3 => sel3,
        z => y(1)
);

mux2: mux16_1bit port map (
        in0 => input(2),
        in1 => input(1),
        in2 => input(0),
        in3 => input(15),
        in4 => input(14),
        in5 => input(13),
        in6 => input(12),
        in7 => input(11),
        in8 => input(10),
        in9 => input(9),
        in10 => input(8),
        in11 => input(7),
        in12 => input(6),
        in13 => input(5),
        in14 => input(4),
        in15 => input(3),
        s0 => sel0,
        s1 => sel1,
        s2 => sel2,
        s3 => sel3,
        z => y(2)
);

mux3: mux16_1bit port map (
        in0 => input(3),
        in1 => input(2),
        in2 => input(1),
        in3=> input(0),
        in4 => input(15),
        in5 => input(14),
        in6 => input(13),
        in7 => input(12),
        in8 => input(11),
        in9 => input(10),
        in10 => input(9),
        in11 => input(8),
		in12 => input(7),
		in13 => input(6),
		in14 => input(5),
		in15 => input(4),
		s0 => sel0,
		s1 => sel1,
		s2 => sel2,
		s3 => sel3,
		z => y(3)
);

mux4: mux16_1bit port map (
		in0 => input(4),
		in1 => input(3),
		in2 => input(2),
		in3 => input(1),
		in4 => input(0),
		in5 => input(15),
		in6 => input(14),
		in7 => input(13),
		in8 => input(12),
		in9 => input(11),
		in10 => input(10),
		in11 => input(9),
		in12 => input(8),
		in13 => input(7),
		in14 => input(6),
		in15 => input(5),
		s0 => sel0,
		s1 => sel1,
		s2 => sel2,
		s3 => sel3,
		z => y(4)
);

mux5: mux16_1bit port map (
		in0 => input(5),
		in1 => input(4),
		in2 => input(3),
		in3 => input(2),
		in4 => input(1),
		in5 => input(0),
		in6 => input(15),
		in7 => input(14),
		in8 => input(13),
		in9 => input(12),
		in10 => input(11),
		in11 => input(10),
		in12 => input(9),
		in13 => input(8),
		in14 => input(7),
		in15 => input(6),
		s0 => sel0,
		s1 => sel1,
		s2 => sel2,
		s3 => sel3,
		z => y(5)
);

mux6: mux16_1bit port map (
		in0 => input(6),
		in1 => input(5),
		in2 => input(4),
		in3 => input(3),
		in4 => input(2),
		in5 => input(1),
        in6 => input(0),
        in7 => input(15),
        in8 => input(14),
        in9 => input(13),
        in10 => input(12),
        in11 => input(11),
        in12 => input(10),
        in13 => input(9),
        in14 => input(8),
        in15 => input(7),
        s0 => sel0,
        s1 => sel1,
        s2 => sel2,
        s3 => sel3,
        z => y(6)
);

mux7: mux16_1bit port map (
        in0 => input(7),
        in1 => input(6),
        in2 => input(5),
        in3 => input(4),
        in4 => input(3),
        in5 => input(2),
        in6 => input(1),
        in7 => input(0),
        in8 => input(15),
        in9 => input(14),
        in10 => input(13),
        in11 => input(12),
        in12 => input(11),
        in13 => input(10),
        in14 => input(9),
        in15 => input(8),
        s0 => sel0,
        s1 => sel1,
        s2 => sel2,
        s3 => sel3,
        z => y(7)
);

mux8: mux16_1bit port map (
        in0 => input(8),
        in1 => input(7),
        in2 => input(6),
        in3 => input(5),
        in4 => input(4),
        in5 => input(3),
        in6 => input(2),
        in7 => input(1),
        in8 => input(0),
        in9 => input(15),
        in10 => input(14),
        in11 => input(13),
        in12 => input(12),
        in13 => input(11),
        in14 => input(10),
        in15 => input(9),
        s0 => sel0,
        s1 => sel1,
        s2 => sel2,
        s3 => sel3,
        z => y(8)
);

mux9: mux16_1bit port map (
		in0 => input(9),
		in1 => input(8),
		in2 => input(7),
		in3 => input(6),
		in4 => input(5),
		in5 => input(4),
		in6 => input(3),
		in7 => input(2),
		in8 => input(1),
		in9 => input(0),
		in10=> input(15),
		in11 => input(14),
		in12 => input(13),
		in13 => input(12),
		in14 => input(11),
		in15 => input(10),
		s0 => sel0,
		s1 => sel1,
		s2 => sel2,
		s3 => sel3,
		z => y(9)
);

mux10: mux16_1bit port map (
		in0 => input(10),
		in1 => input(9),
		in2 => input(8),
		in3 => input(7),
		in4 => input(6),
		in5 => input(5),
		in6 => input(4),
		in7 => input(3),
		in8 => input(2),
		in9 => input(1),
		in10 => input(0),
		in11 => input(15),
		in12 => input(14),
		in13 => input(13),
		in14 => input(12),
		in15 => input(11),
		s0 => sel0,
		s1 => sel1,
		s2 => sel2,
		s3 => sel3,
		z => y(10)
);

mux11: mux16_1bit port map (
		in0 => input(11),
		in1 => input(10),
		in2 => input(9),
		in3 => input(8),
		in4 => input(7),
		in5 => input(6),
		in6 => input(5),
		in7 => input(4),
		in8 => input(3),
		in9 => input(2),
		in10 => input(1),
		in11 => input(0),
		in12 => input(15),
		in13 => input(14),
		in14 => input(13),
		in15 => input(12),
		s0 => sel0,
		s1 => sel1,
        s2 => sel2,
        s3 => sel3,
        z => y(11)
);

mux12: mux16_1bit port map (
        in0 => input(12),
        in1 => input(11),
        in2 => input(10),
        in3 => input(9),
        in4 => input(8),
        in5 => input(7),
        in6 => input(6),
        in7 => input(5),
        in8 => input(4),
        in9 => input(3),
        in10 => input(2),
        in11 => input(1),
        in12 => input(0),
        in13 => input(15),
        in14 => input(14),
        in15 => input(13),
        s0 => sel0,
        s1 => sel1,
        s2 => sel2,
        s3 => sel3,
        z => y(12)
);
mux13: mux16_1bit port map (
        in0 => input(13),
        in1 => input(12),
        in2 => input(11),
        in3 => input(10),
        in4 => input(9),
        in5 => input(8),
        in6 => input(7),
        in7 => input(6),
        in8 => input(5),
        in9 => input(4),
        in10 => input(3),
        in11 => input(2),
        in12 => input(1),
        in13 => input(0),
        in14 => input(15),
        in15 => input(14),
        s0 => sel0,
        s1 => sel1,
        s2 => sel2,
        s3 => sel3,
        z => y(13)
);

mux14: mux16_1bit port map (
        in0 => input(14),
        in1 => input(13),
        in2 => input(12),
        in3 => input(11),
        in4 => input(10),
        in5 => input(9),
        in6 => input(8),
        in7 => input(7),
        in8 => input(6),
        in9 => input(5),
        in10 => input(4),
        in11 => input(3),
        in12 => input(2),
                        in13 => input(1),
                        in14 => input(0),
                        in15 => input(15),
                        s0 => sel0,
                        s1 => sel1,
                        s2 => sel2,
                        s3 => sel3,
                        z => y(14)
                );

                mux15: mux16_1bit port map (
                        in0 => input(15),
                        in1 => input(14),
                        in2 => input(13),
                        in3 => input(12),
                        in4 => input(11),
                        in5 => input(10),
                        in6 => input(9),
                        in7 => input(8),
                        in8 => input(7),
                        in9 => input(6),
                        in10 => input(5),
                        in11 => input(4),
                        in12 => input(3),
                        in13 => input(2),
                        in14 => input(1),
                        in15 => input(0),
                        s0 => sel0,
                        s1 => sel1,
                        s2 => sel2,
                        s3 => sel3,
                        z => y(15)
                );
        end Behavioral;
```

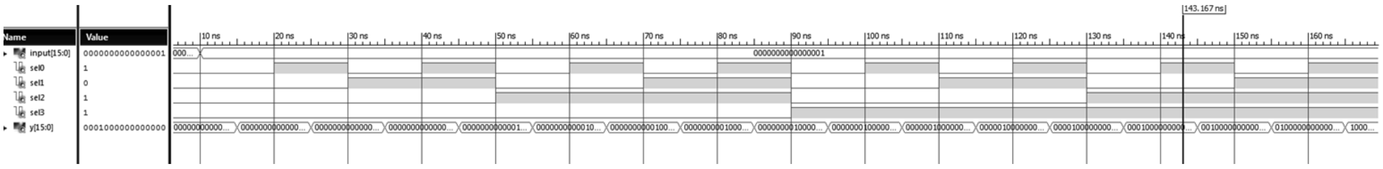
2.4.2 Shifter_16bit Test Bench

```
LIBRARY ieee;
  USE ieee.std_logic_1164.ALL;
  USE ieee.numeric_std.ALL;

  ENTITY testbench IS
  END testbench;

  ARCHITECTURE behavior OF testbench IS

  component shifter_16bit
  Port ( input: in STD_LOGIC_VECTOR(15 downto 0);
          sel0,sel1,sel2,sel3 : in  STD_LOGIC;
          y: out STD_LOGIC_VECTOR(15 downto 0) );
  end component;

  -- SIGNALS
 signal input: std_logic_vector(15 downto 0) := (others => '0');
 signal sel0,sel1,sel2,sel3 : STD_LOGIC := '0';
 signal y: std_logic_vector(15 downto 0);

  BEGIN
        uut: shifter_16bit PORT MAP(
                        input => input,
                        sel0 => sel0,
                        sel1 => sel1,
                        sel2 => sel2,
                        sel3 => sel3,
                        y => y
        );
```

```
tb : PROCESS
BEGIN
                        wait for 10 ns;
                        input <= "0000000000000001";
                        sel0 <= '0';
                        sel1 <= '0';
                        sel2 <= '0';
                        sel3 <= '0';

                        wait for 10 ns;
                        sel0 <= '1';
                        sel1 <= '0';
                        sel2 <= '0';
                        sel3 <= '0';

                        wait for 10 ns;
                        sel0 <= '0';
                        sel1 <= '1';
                        sel2 <= '0';
                        sel3 <= '0';

                        wait for 10 ns;
                        sel0 <= '1';
                        sel1 <= '1';
                        sel2 <= '0';
                        sel3 <= '0';

                        wait for 10 ns;
                        sel0 <= '0';
                        sel1 <= '0';
                        sel2 <= '1';
                        sel3 <= '0';

                        wait for 10 ns;
                        sel0 <= '1';
                        sel1 <= '0';
                        sel2 <= '1';
                        sel3 <= '0';

                        wait for 10 ns;
                        sel0 <= '0';
                        sel1 <= '1';
                        sel2 <= '1';
                        sel3 <= '0';

                        wait for 10 ns;
                        sel0 <= '1';
                        sel1 <= '1';
                        sel2 <= '1';
                        sel3 <= '0';

                        wait for 10 ns;
                        sel0 <= '0';
                        sel1 <= '0';
                        sel2 <= '0';
                        sel3 <= '1';

                        wait for 10 ns;
                        sel0 <= '1';
                        sel1 <= '0';
                        sel2 <= '0';
                        sel3 <= '1';

                        wait for 10 ns;
```

```
                                        sel0 <= '0';
                                        sel1 <= '1';
                                        sel2 <= '0';
                                        sel3 <= '1';

                                        wait for 10 ns;
                                        sel0 <= '1';
                                        sel1 <= '1';
                                        sel2 <= '0';
                                        sel3 <= '1';

                                        wait for 10 ns;
                                        sel0 <= '0';
                                        sel1 <= '0';
                                        sel2 <= '1';
                                        sel3 <= '1';

                                        wait for 10 ns;
                                        sel0 <= '1';
                                        sel1 <= '0';
                                        sel2 <= '1';
                                        sel3 <= '1';

                                        wait for 10 ns;
                                        sel0 <= '0';
                                        sel1 <= '1';
                                        sel2 <= '1';
                                        sel3 <= '1';

                                        wait for 10 ns;
                                        sel0 <= '1';
                                        sel1 <= '1';
                                        sel2 <= '1';
                                        sel3 <= '1';
                                        wait;
  END PROCESS tb;
 END;
```

2.4.3 Shifter_16bit Test Results



2.5 Full_adder

2.5.1 Full_adder Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity full_adder is
    Port ( A: in  STD_LOGIC;
           B: in  STD_LOGIC;
           c_in: in  STD_LOGIC;
           c_out: out  STD_LOGIC;
           Sum: out  STD_LOGIC);
end full_adder;
```

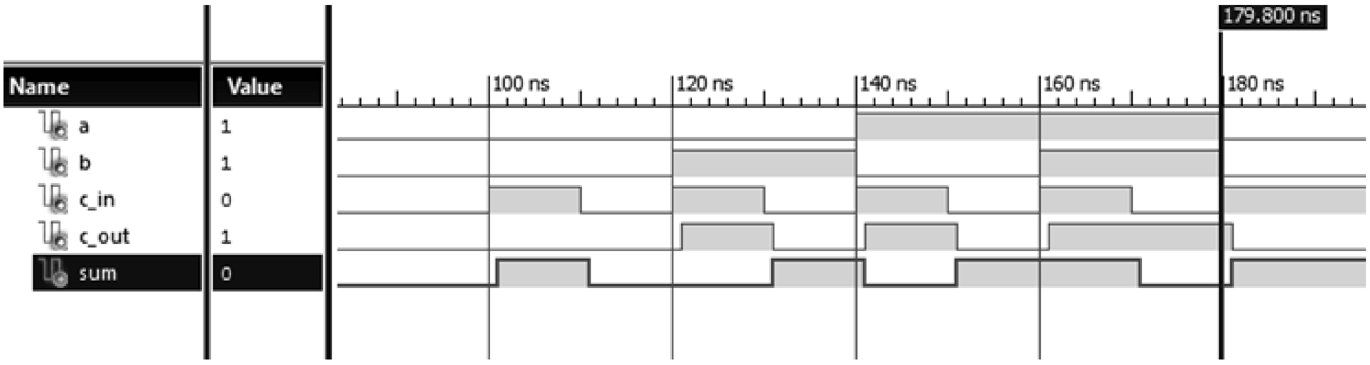
```
architecture Behavioral of full_adder is

begin
c_out     <=  (A and B) or ((A xor B) and c_in)after 1 ns;
Sum       <=  ((A xor B) xor c_in)                 after 1 ns;

end Behavioral;
```

2.5.2 Full_adder Test Bench

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY full_adder_tb IS
END full_adder_tb;

ARCHITECTURE behavior OF full_adder_tb IS

   -- Component Declaration for the Unit Under Test (UUT)
   COMPONENT full_adder
   PORT(
        A : IN  std_logic;
        B : IN  std_logic;
        c_in : IN  std_logic;
        c_out : OUT  std_logic;
        Sum : OUT  std_logic
       );
   END COMPONENT;

  --Inputs
  signal A : std_logic := '0';
  signal B : std_logic := '0';
  signal c_in : std_logic := '0';
  --Outputs
  signal c_out : std_logic;
  signal Sum : std_logic;

BEGIN
  uut: full_adder PORT MAP (
         A => A,
         B => B,
         c_in => c_in,
         c_out => c_out,
         Sum => Sum
  );

  stim_proc: process
  begin
          wait for 100 ns;
          A <= '0';
          B <= '0';
          c_in <= '1';

          --loop through A values
          for i in 0 to 1 loop
                    --loop through B values
                    for j in 0 to 1 loop
```

```
                        --loop through c_in values
                        for k in 0 to 1 loop
                                wait for 10 ns;
                                c_in <= not c_in;
                        end loop;
                        B <= not B;
                end loop;
                A <= not A;
        end loop;

        wait;
  end process;

END;
```

2.5.3 Full_adder Test Results



2.6 Ripple_adder_16bit

2.6.1 Ripple_adder_16bit Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity ripple_adder_16bit is
   Port ( Ain, Bin : in  STD_LOGIC_VECTOR (15 downto 0);
          C_in : in  STD_LOGIC; -- TODO: should this be A_in?
          C_out : out  STD_LOGIC;
          Z : out  STD_LOGIC_VECTOR (15 downto 0));
end ripple_adder_16bit;

architecture Behavioral of ripple_adder_16bit is
-- COMPONENTS
        component full_adder is
                Port (  A: in  STD_LOGIC;
                        B: in  STD_LOGIC;
                        c_in: in  STD_LOGIC;
                        c_out: out  STD_LOGIC;
                        Sum: out  STD_LOGIC);
        end component;

-- SIGNALS
signal c1, c2, c3, c4, c5, c6, c7, c8, c9, c10, c11, c12, c13, c14, c15: STD_LOGIC;
```

```
begin
-- PORT MAPS
        full0: full_adder port map (
                A => Ain(0),
                B => Bin(0),
                c_in => C_in,
                c_out => c1,
                Sum => Z(0)
        );

        full1: full_adder port map (
                A => Ain(1),
                B => Bin(1),
                c_in => c1,
                c_out => c2,
                Sum => Z(1)
        );

        full2: full_adder port map (
                A => Ain(2),
                B => Bin(2),
                c_in => c2,
                c_out => c3,
                Sum => Z(2)
        );

        full3: full_adder port map (
                A => Ain(3),
                B => Bin(3),
                c_in => c3,
                c_out => c4,
                Sum => Z(3)
        );

        full4: full_adder port map (
                A => Ain(4),
                B => Bin(4),
                c_in => c4,
                c_out => c5,
                Sum => Z(4)
        );

        full5: full_adder port map (
                A => Ain(5),
                B => Bin(5),
                c_in => c5,
                c_out => c6,
                Sum => Z(5)
        );

        full6: full_adder port map (
                A => Ain(6),
                B => Bin(6),
                c_in => c6,
                c_out => c7,
                Sum => Z(6)
        );
```

```
full7: full_adder port map (
        A => Ain(7),
        B => Bin(7),
        c_in => c7,
        c_out => c8,
        Sum => Z(7)
);

full8: full_adder port map (
        A => Ain(8),
        B => Bin(8),
        c_in => c8,
        c_out => c9,
        Sum => Z(8)
);

full9: full_adder port map (
        A => Ain(9),
        B => Bin(9),
        c_in => c9,
        c_out => c10,
        Sum => Z(9)
);

full10: full_adder port map (
        A => Ain(10),
        B => Bin(10),
        c_in => c10,
        c_out => c11,
        Sum => Z(10)
);

full11: full_adder port map (
        A => Ain(11),
        B => Bin(11),
        c_in => c11,
        c_out => c12,
        Sum => Z(11)
);

full12: full_adder port map (
        A => Ain(12),
        B => Bin(12),
        c_in => c12,
        c_out => c13,
        Sum => Z(12)
);

full13: full_adder port map (
        A => Ain(13),
        B => Bin(13),
        c_in => c13,
        c_out => c14,
        Sum => Z(13)
);
```

```
		full14: full_adder port map (
				A => Ain(14),
				B => Bin(14),
				c_in => c14,
				c_out => c15,
				Sum => Z(14)
		);

		full15: full_adder port map (
				A => Ain(15),
				B => Bin(15),
				c_in => c15,
				c_out => C_out,
				Sum => Z(15)
		);

end Behavioral;
```

2.6.2 Ripple_adder_16bit Test Bench

2.6.3 Ripple_adder_16bit Test Results

2.7 Arithmetic_logic_unit

2.7.1 Arithmetic_logic_unit Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

entity arithmetic_logic_unit is
    Port ( dataA, dataB : in  STD_LOGIC_VECTOR (15 downto 0);
           FS_code : in  STD_LOGIC_VECTOR (4 downto 0);
           V, C_out : out  STD_LOGIC;
           dataG : out  STD_LOGIC_VECTOR (15 downto 0));
end arithmetic_logic_unit;

architecture Behavioral of arithmetic_logic_unit is

begin
		--assign dataG
dataG <=  dataA                                          after 5 ns when FS_code="00000" else
   std_logic_vector( unsigned(dataA) + 1)       after 5 ns when FS_code="00001" else
   std_logic_vector( unsigned(dataA) + unsigned(dataB))              after 5 ns when FS_code="00010" else
   std_logic_vector( unsigned(dataA) + unsigned(dataB) + 1)      after 5 ns when FS_code="00011" else
   std_logic_vector( unsigned(dataA) + unsigned(not dataB))      after 5 ns when FS_code="00100" else
   std_logic_vector( unsigned(dataA) + unsigned(not dataB) + 1)       after 5 ns when FS_code="00101"
             else
   std_logic_vector( unsigned(dataA) - 1)       after 5 ns when FS_code="00110" else
   dataA                                        after 5 ns when FS_code="00111" else
   dataA and dataB                              after 5 ns when FS_code="01000" else
   dataA or dataB                               after 5 ns when FS_code="01010" else
   dataA xor dataB                    after 5 ns when FS_code="01100" else
   not dataA                                    after 5 ns when FS_code="01110";
```

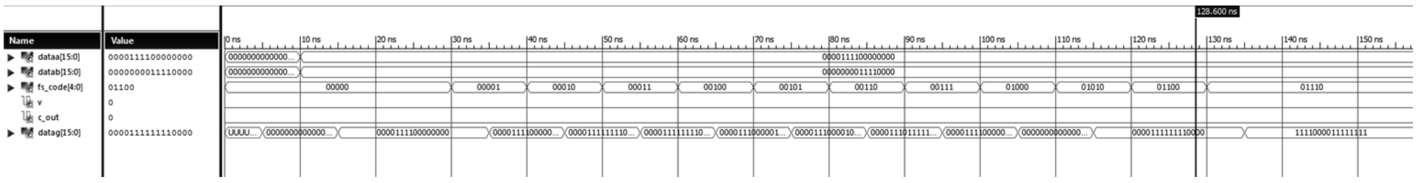
```
--assign flags
V <= '0';
C_out <= '0';

end Behavioral;
```

2.7.2 Arithmetic_logic_unit Test Bench

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY ALU_tb IS
END ALU_tb;

ARCHITECTURE behavior OF ALU_tb IS

  COMPONENT arithmetic_logic_unit
  PORT(           dataA, dataB : IN  std_logic_vector(15 downto 0);
                  FS_code : IN  std_logic_vector(4 downto 0);
                  V, C_out : OUT  std_logic;
                  dataG : OUT  std_logic_vector(15 downto 0) );
  END COMPONENT;

  --Inputs
  signal dataA : std_logic_vector(15 downto 0) := (others => '0');
  signal dataB : std_logic_vector(15 downto 0) := (others => '0');
  signal FS_code : std_logic_vector(4 downto 0) := (others => '0');
        --Outputs
  signal V : std_logic;
  signal C_out : std_logic;
  signal dataG : std_logic_vector(15 downto 0);

BEGIN
  uut: arithmetic_logic_unit PORT MAP (
      dataA => dataA,
      dataB => dataB,
      FS_code => FS_code,
      V => V,
      C_out => C_out,
      dataG => dataG
     );

  stim_proc: process
  begin
        wait for 10 ns;
        dataA <= "0000111100000000";
        dataB <= "0000000011110000";

        wait for 10 ns;     FS_code <= "00000";
        wait for 10 ns;     FS_code <= "00001";
        wait for 10 ns;     FS_code <= "00010";
        wait for 10 ns;     FS_code <= "00011";
        wait for 10 ns;     FS_code <= "00100";
        wait for 10 ns;     FS_code <= "00101";
        wait for 10 ns;     FS_code <= "00110";
        wait for 10 ns;     FS_code <= "00111";
```

```
                wait for 10 ns;      FS_code <= "01000";
                wait for 10 ns;      FS_code <= "01010";
                wait for 10 ns;      FS_code <= "01100";
                wait for 10 ns;      FS_code <= "01110";
                wait;
        end process;

    END;
```

2.7.3 Arithmetic_logic_unit Test Results



2.8 Function_unit

2.8.1 Function_unit Code

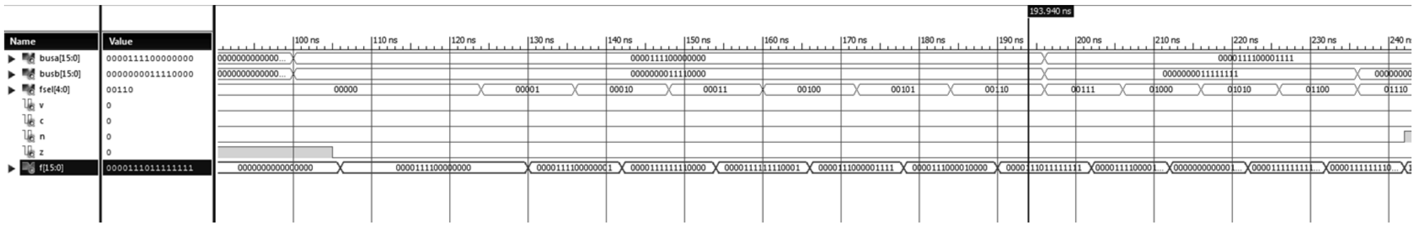
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity function_unit is
Port ( busA, busB : in  STD_LOGIC_VECTOR (15 downto 0);
       FSel : in  STD_LOGIC_VECTOR (4 downto 0);
       V, C, N, Z : out  STD_LOGIC;
       F : out  STD_LOGIC_VECTOR (15 downto 0));
end function_unit;

architecture Behavioral of function_unit is
-- COMPONENTS
        component shifter_16bit is
        Port(   input: in STD_LOGIC(15 to 0);
                sel0,sel1,sel2,sel3 : in  STD_LOGIC;
                y: out STD_LOGIC(15 to 0));
        end component;

        component zero_detect_16 is
         Port (  data : in  STD_LOGIC_VECTOR (15 downto 0);
                 z : out  STD_LOGIC);
        end component;

        component mux2_16bit is
         Port (   S: in  STD_LOGIC;
                  In0, In1: in  STD_LOGIC_VECTOR (15 downto 0);
                  Z: out  STD_LOGIC_VECTOR (15 downto 0));
        end component;

        component arithmetic_logic_unit is
         Port (  busA, busB : in  STD_LOGIC_VECTOR (15 downto 0);
                 G_sel : in  STD_LOGIC_VECTOR (4 downto 0);
                 V, C : out  STD_LOGIC;
                 G : out  STD_LOGIC_VECTOR (15 downto 0));
        end component;

        -- SIGNALS
        signal ALU_out, shifter_out: STD_LOGIC_VECTOR (15 downto 0);

begin
--PORT DECLARATIONS
        ALU: arithmetic_logic_unit port map (
                dataA => busA,
                dataB => busB,
                FS_code => FSel,
                V => V,
                C_out => C,
                dataG => ALU_out
        );

        shifter: shifter_16bit port map (
                input => busB,
                sel0 => FSel(0),
                sel1 => FSel(1),
                sel2 => FSel(2),
                sel3 => FSel(3),
                y => shifter_out
        );

        z_flag: zero_detect_16 port map (
                data => ALU_out,
                z => Z
        );

        muxF: mux2_16bit port map (
                S => FSel(4),
                In0 => ALU_out,
                In1 => shifter_out,
                Z => F
        );

        N <= output(15);
        F <= output;
end Behavioral;
```

2.8.2 Function_unit Test Bench

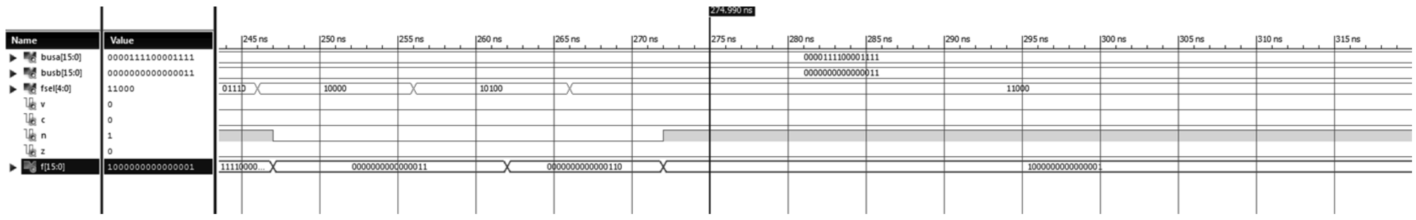
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY function_unit_tb IS
END function_unit_tb;

ARCHITECTURE behavior OF function_unit_tb IS
   COMPONENT function_unit
   PORT ( busA, busB : in  STD_LOGIC_VECTOR (15 downto 0);
         FSel : in  STD_LOGIC_VECTOR (4 downto 0);
         V, C, N, Z : out  STD_LOGIC;
         F : out  STD_LOGIC_VECTOR (15 downto 0));
   END COMPONENT;
   --Inputs
   signal busA, busB : std_logic_vector(15 downto 0) := (others => '0');
   signal FSel : std_logic_vector(4 downto 0) := (others => '0');
 --Outputs
   signal V, C, N, Z : std_logic;
   signal F : std_logic_vector(15 downto 0);

BEGIN
   uut: function_unit PORT MAP (
          busA => busA,
          busB => busB,
          FSel => FSel,
          V => V,
          C => C,
          N => N,
          Z => Z,
          F => F
        );

   stim_proc: process
   begin
              wait for 100 ns;

              --Test arithmetic functions
              busA <= "0000111100000000";
              busB <= "0000000011110000";
              wait for 10 ns;      FSel <= "00000";
              wait for 10 ns;      FSel <= "00001";
              wait for 10 ns;      FSel <= "00010";
              wait for 10 ns;      FSel <= "00011";
              wait for 10 ns;      FSel <= "00100";
              wait for 10 ns;      FSel <= "00101";
              wait for 10 ns;      FSel <= "00110";
              wait for 10 ns;      FSel <= "00111";

              --Test logical functions
              busA <= "0000111100001111";
              busB <= "0000000011111111";
              wait for 10 ns;      FSel <= "01000";
              wait for 10 ns;      FSel <= "01010";
              wait for 10 ns;      FSel <= "01100";
              wait for 10 ns;      FSel <= "01110";

              --Test shifter
              busB <= "0000000000000011";
              wait for 10 ns;      FSel <= "10000";
              wait for 10 ns;      FSel <= "10100";
              wait for 10 ns;      FSel <= "11000";
              wait;
   end process;
END;
```

2.8.3 Function_unit Test Results

Arithmetic & Logic testing



Shift testing



3 Datapath

3.1 Datapath Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

entity datapath is
    Port ( data_in, constant_in : in  STD_LOGIC_VECTOR (15 downto 0);
           dest_select, Aselect, Bselect : in  STD_LOGIC_VECTOR (2 downto 0);
           FSelect : in  STD_LOGIC_VECTOR (4 downto 0);
           load_enable, CLK, MBSelect, MDSelect : in  STD_LOGIC;
           data_out, address_out : out  STD_LOGIC_VECTOR (15 downto 0);
           V,C,N,Z : out  STD_LOGIC);
end datapath;

architecture Behavioral of datapath is
--COMPONENTS
              component register_file_v2 is
              port(  load_enable, CLK : in  STD_LOGIC;
          dest_select, Asel, Bsel : in  STD_LOGIC_VECTOR(2 downto 0);
          DData : in  STD_LOGIC_VECTOR (15 downto 0);
          AData, BData : out  STD_LOGIC_VECTOR (15 downto 0));
              end component;

              component function_unit is
   port(  busA, busB : in  STD_LOGIC_VECTOR (15 downto 0);
          FSel : in  STD_LOGIC_VECTOR (4 downto 0);
          V, C, N, Z : out  STD_LOGIC;
          F : out  STD_LOGIC_VECTOR (15 downto 0));
              end component;

              component mux2_16bit is
              port(  S: in  STD_LOGIC;
          In0, In1: in  STD_LOGIC_VECTOR (15 downto 0);
```

```
         Z: out  STD_LOGIC_VECTOR (15 downto 0)
       );
   end component;

--SIGNALS
signal Bdata, FData, ABus, BBus, MuxDData: STD_LOGIC_VECTOR (15 downto 0);

begin
--PORT MAPS
        regFile: register_file_v2 port map (
                load_enable => load_enable,
                CLK => CLK,
                dest_select => dest_select,
                Asel => Aselect,
                Bsel => Bselect,
                DData => MuxDData,
                AData => ABus,
                BData => Bdata
        );

        functUnit: function_unit port map (
                busA => ABus,
                busB => BBus,
                FSel => FSelect,
                V => V,
                C => C,
                N => N,
                Z => Z,
                F => FData
        );

        muxB: mux2_16bit port map (
                S => MBSelect,
                In0 => Bdata,
                In1 => constant_in,
                Z => BBus
        );

        muxD: mux2_16bit port map (
                S => MDSelect,
                In0 => FData,
                In1 => data_in,
                Z => MuxDData
        );

        data_out <= BBus;
        address_out <= ABus;
end Behavioral;
```

3.2 Datapath Test Bench

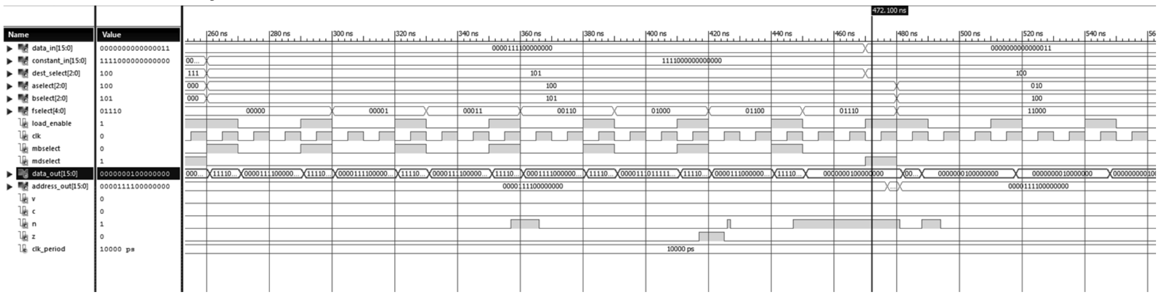
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
use ieee.numeric_std.all;

ENTITY datapath_tb IS
END datapath_tb;
```

```
ARCHITECTURE behavior OF datapath_tb IS
    COMPONENT datapath
    PORT(           data_in, constant_in : in  STD_LOGIC_VECTOR (15 downto 0);
                    dest_select, Aselect, Bselect : in  STD_LOGIC_VECTOR (2 downto 0);
                    FSelect : in  STD_LOGIC_VECTOR (4 downto 0);
                    load_enable, CLK, MBSelect, MDSelect : in  STD_LOGIC;
                    data_out, address_out : out  STD_LOGIC_VECTOR (15 downto 0);
                    V,C,N,Z : out  STD_LOGIC);
    END COMPONENT;

   --Inputs
   signal data_in, constant_in : std_logic_vector(15 downto 0) := (others => '0');
   signal dest_select, Aselect, Bselect : std_logic_vector(2 downto 0) := (others => '0');
   signal FSelect : std_logic_vector(4 downto 0) := (others => '0');
   signal load_enable, CLK, MBSelect, MDSelect : std_logic := '0';
   --Outputs
   signal data_out, address_out : std_logic_vector(15 downto 0);
   signal V,C,N,Z : std_logic;

   constant CLK_period : time := 10 ns;

BEGIN
   uut: datapath PORT MAP (
          data_in => data_in,
          constant_in => constant_in,
          dest_select => dest_select,
          Aselect => Aselect,
          Bselect => Bselect,
          FSelect => FSelect,
          load_enable => load_enable,
          CLK => CLK,
          MBSelect => MBSelect,
          MDSelect => MDSelect,
          data_out => data_out,
          address_out => address_out,
          V => V,
          C => C,
          N => N,
          Z => Z
        );

   CLK_process :process
   begin
                CLK <= '0';
                wait for CLK_period/2;
                CLK <= '1';
                wait for CLK_period/2;
   end process;

   stim_proc: process
   begin
      wait for 100 ns;

                --ALU TEST DESCRIPTION:
                -- This test loads the value "0000111100000000" into every register. This value is read from
                -- register 4 into the function unit (through data line A. Each function is tested and the result
                -- stored into register 5. The contents of register 4 can be read on address_out. When
                -- MBSelect is set to '1', register 5 can be read on data_out.

                --Set Registers
                MDSelect <= '1';
                data_in <= "0000111100000000";
                dest_select <= "000";
                load_enable <= '1';

                for i in 0 to 7 loop
                        wait for CLK_period*2;
                        dest_select <= STD_LOGIC_VECTOR (unsigned(dest_select) + 1);
                end loop;

                --Test ALU functionality as described above
                MBSelect <= '1';
                MDSelect <= '0';
                dest_select <= "101";
                Aselect <= "100";
                Bselect <= "101";
                constant_in <= "1111000000000000";

                MBSelect <= '1';
                load_enable <= '1';                 wait for 10 ns;
                FSelect <= "00000";        -- A
                load_enable <= '0';
                MBSelect <= '0';   wait for 20 ns;

                MBSelect <= '1';
                load_enable <= '1';                 wait for 10 ns;
                FSelect <= "00001";        -- A + 1
                load_enable <= '0';
                MBSelect <= '0';   wait for 20 ns;

                MBSelect <= '1';
                load_enable <= '1';                 wait for 10 ns;
                FSelect <= "00011";        -- A + constant + 1
                load_enable <= '0';
                MBSelect <= '0';   wait for 20 ns;

                MBSelect <= '1';
                load_enable <= '1';                 wait for 10 ns;
                FSelect <= "00110";        -- A - 1
                load_enable <= '0';
                MBSelect <= '0';   wait for 20 ns;

                MBSelect <= '1';
                load_enable <= '1';                 wait for 10 ns;
                FSelect <= "01000";        -- A and constant
                load_enable <= '0';
                MBSelect <= '0';   wait for 20 ns;

                MBSelect <= '1';
                load_enable <= '1';                 wait for 10 ns;
                FSelect <= "01100";        -- A xor constant
                load_enable <= '0';
                MBSelect <= '0';   wait for 20 ns;
```

```
                    MBSelect <= '1';
                    load_enable <= '1';                 wait for 10 ns;
                    FSelect <= "01110";         -- not A
                    load_enable <= '0';
                    MBSelect <= '0';   wait for 20 ns;

                    --SHIFT TEST DESCRIPTION:
                    -- This test loads "0000000000000011" into register 4. This value is read from here into the
                    -- function unit (through data line B. The value is shifted and stored back into register 2.
                    -- The contents of register 2 can be read on address_out.

                    --Set Register 4
                    load_enable <= '0';
                    MDSelect <= '1';
                    data_in <= "0000000000000011";
                    dest_select <= "100";
                    load_enable <= '1';         wait for 10 ns;

                    load_enable <= '0';
                    MDSelect <= '0';
                    MBSelect <= '0';
                    Aselect <= "010";
                    Bselect <= "100";
                    FSelect <= "11000";

                    for i in 0 to 15 loop
                         load_enable <= '1';         wait for 10 ns;
                         load_enable <= '0';         wait for 20 ns;
                    end loop;
        wait;
    end process;

END;
```

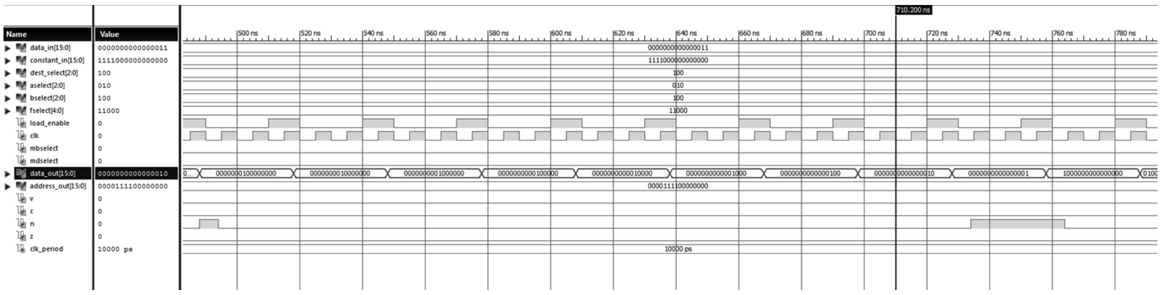
3.3 Datapath Test Results

ALU functionality



Name	Value
data_in[15:0]	0000000000000011
constant_in[15:0]	1111000000000000
dest_select[2:0]	100
aselect[2:0]	100
bselect[2:0]	101
fselect[4:0]	01110
load_enable	1
clk	0
mbselect	0
mdselect	1
data_out[15:0]	0000000100000000
address_out[15:0]	0000111100000000
v	0
c	0
n	1
z	0
clk_period	10000 ps

Shifter Functionality



Name	Value
data_in[15:0]	0000000000000011
constant_in[15:0]	1111000000000000
dest_select[2:0]	100
aselect[2:0]	010
bselect[2:0]	100
fselect[4:0]	11000
load_enable	0
clk	0
mbselect	0
mdselect	0
data_out[15:0]	0000000000000010
address_out[15:0]	0000111100000000
v	0
c	0
n	0
z	0
clk_period	10000 ps